Faculty of Engineering & Technology
Electrical & Computer Engineering Department

ENEE33203

COMPUTER NETWORKS Project #1

Student Name : Shahd Yahya
Student ID : 1210249

Instructor: Dr.Abdalkarim Awad
Section: 2

Part 1:

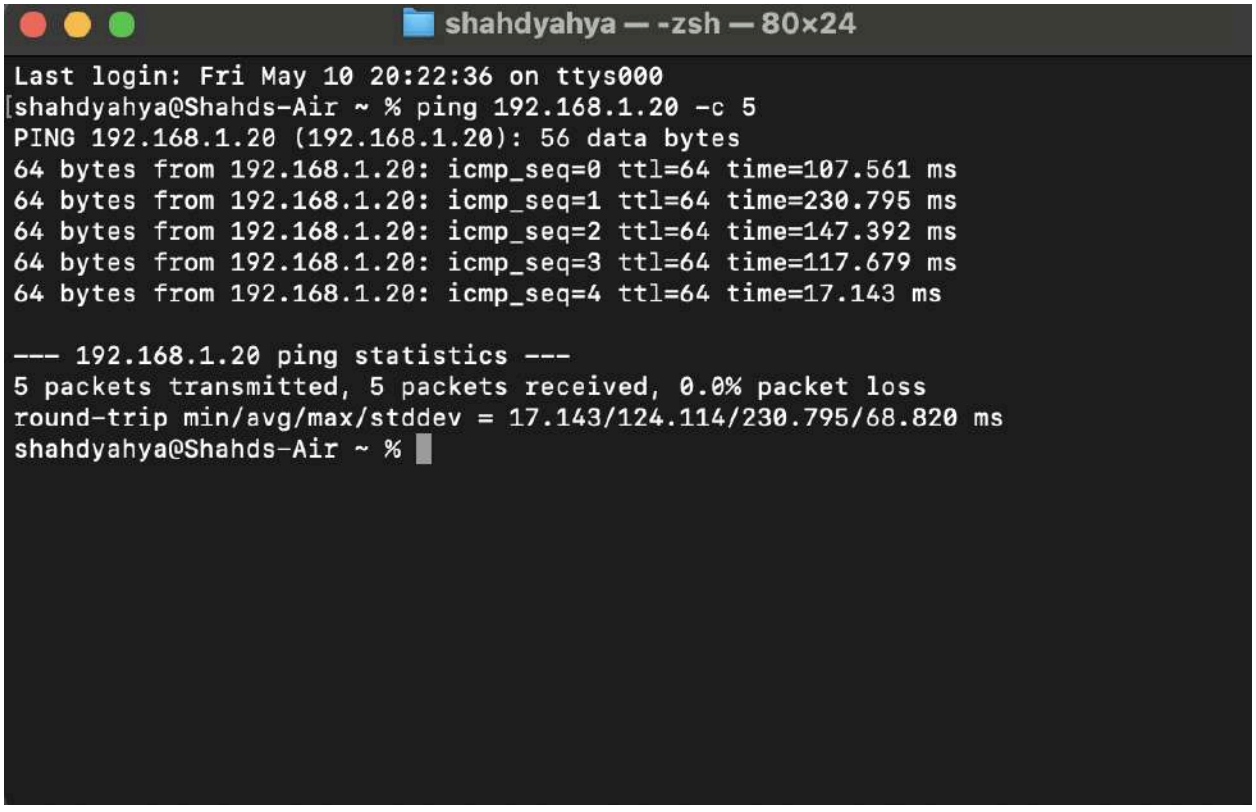
1. **Ping**: It's a simple utility that sends a data packet to another device on the network and waits for a response. This helps check if the device is reachable and how long it takes for the signal to go back and forth (round-trip time).

   Tracert: It tracks the route data packets take from your computer to a destination, showing each "hop" (router) along the way. This can help identify bottlenecks or connection issues along the path.

   Nslookup: It's a tool used to query Domain Name System (DNS) servers. You can use it to translate website addresses (domain names) into numerical IP addresses that computers use, or look up other DNS records.

2.

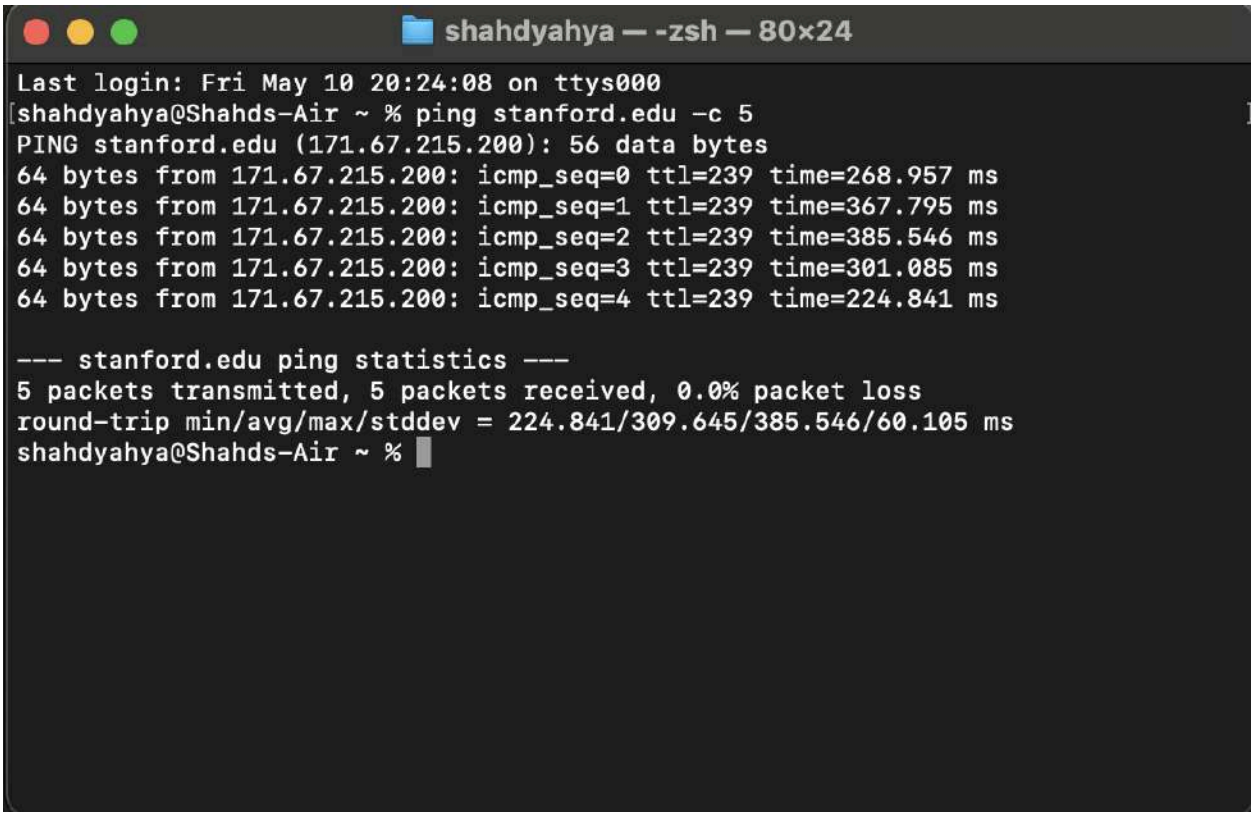
Ping my phone from my laptop:



I have transmitted 5 packets to my phone (in the same network) and received 5 packets with 0% packet loss: All 5 ping requests were sent and successfully received, meaning there were no dropped packets.Round-trip time (RTT) is the total time it takes for a ping request to reach the device and return a response. Here,

the RTT ranged from a minimum of 17.143 milliseconds to a maximum of 230.795 ms, with an average of 124.114 ms.

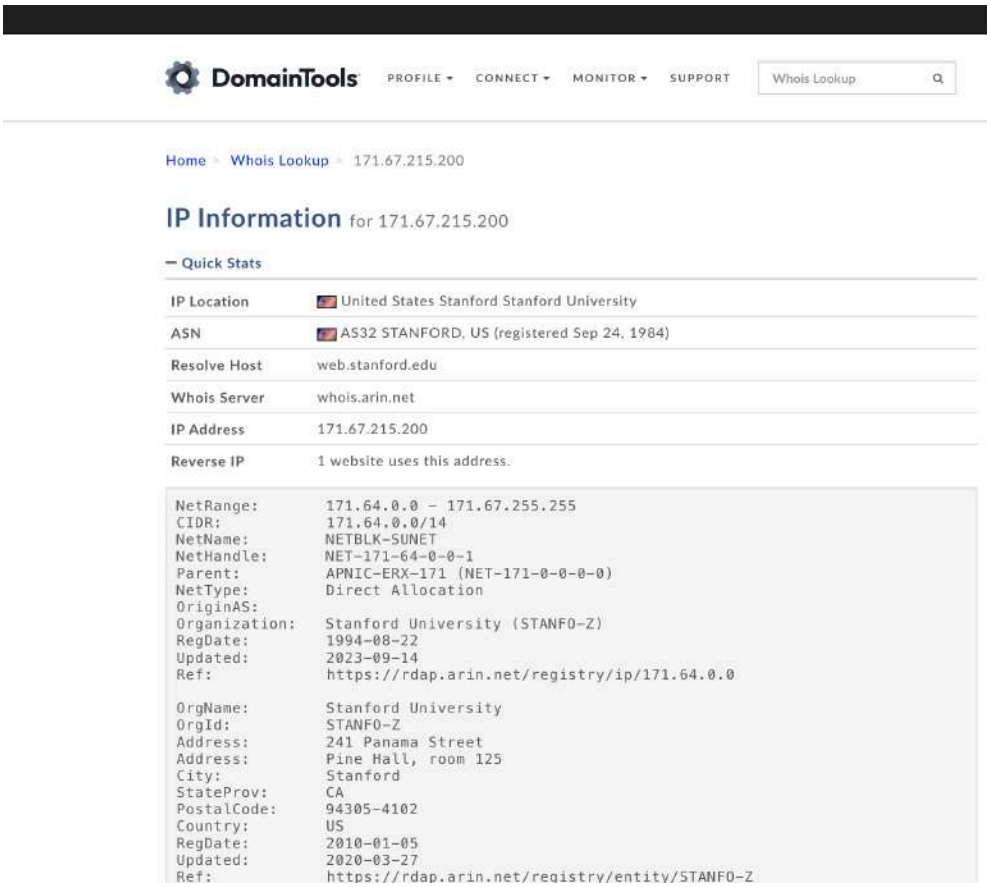
Ping stanford.edu:



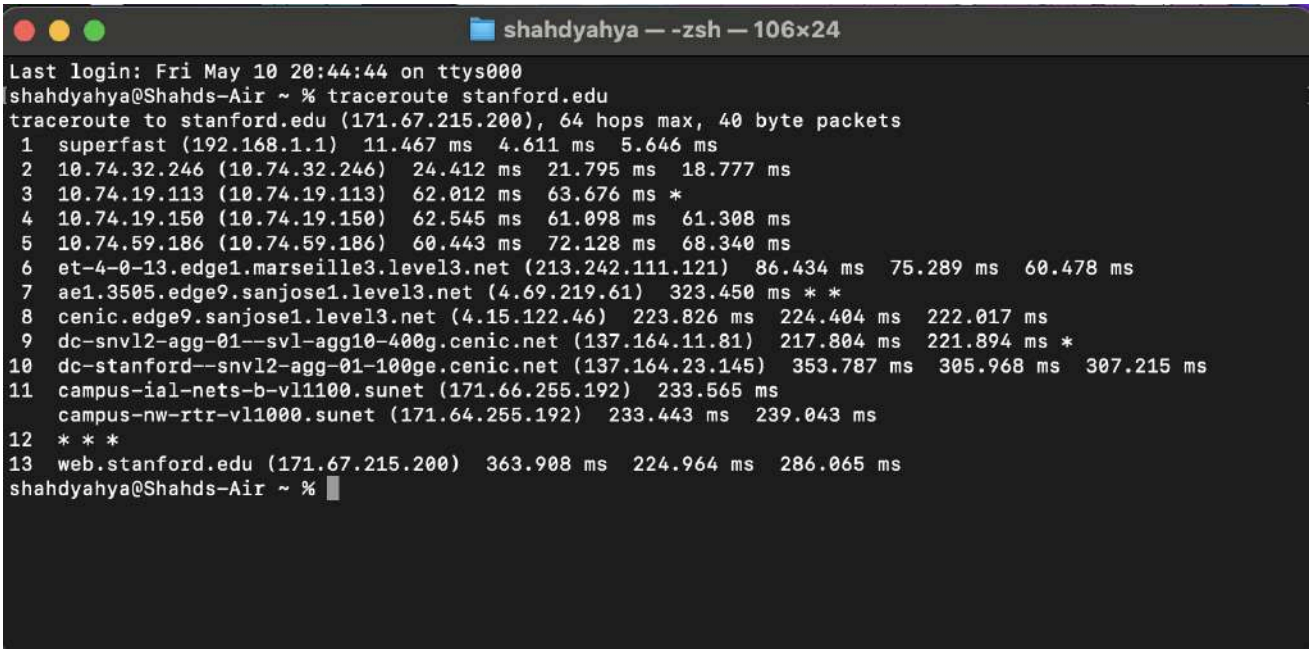
I have transmitted 5 packets and received 5 packets with 0% packet loss: All 5 ping requests were sent and successfully received, meaning there were no dropped packets.Round-trip time (RTT) is the total time it takes for a ping request to reach the device and return a response. Here, the RTT ranged from a minimum of 244.841 ms to a maximum of 385.546 ms, with an average of 309.645 ms.

I have noticed that pinging a device in the same network is faster and that is because of the distance and congestion

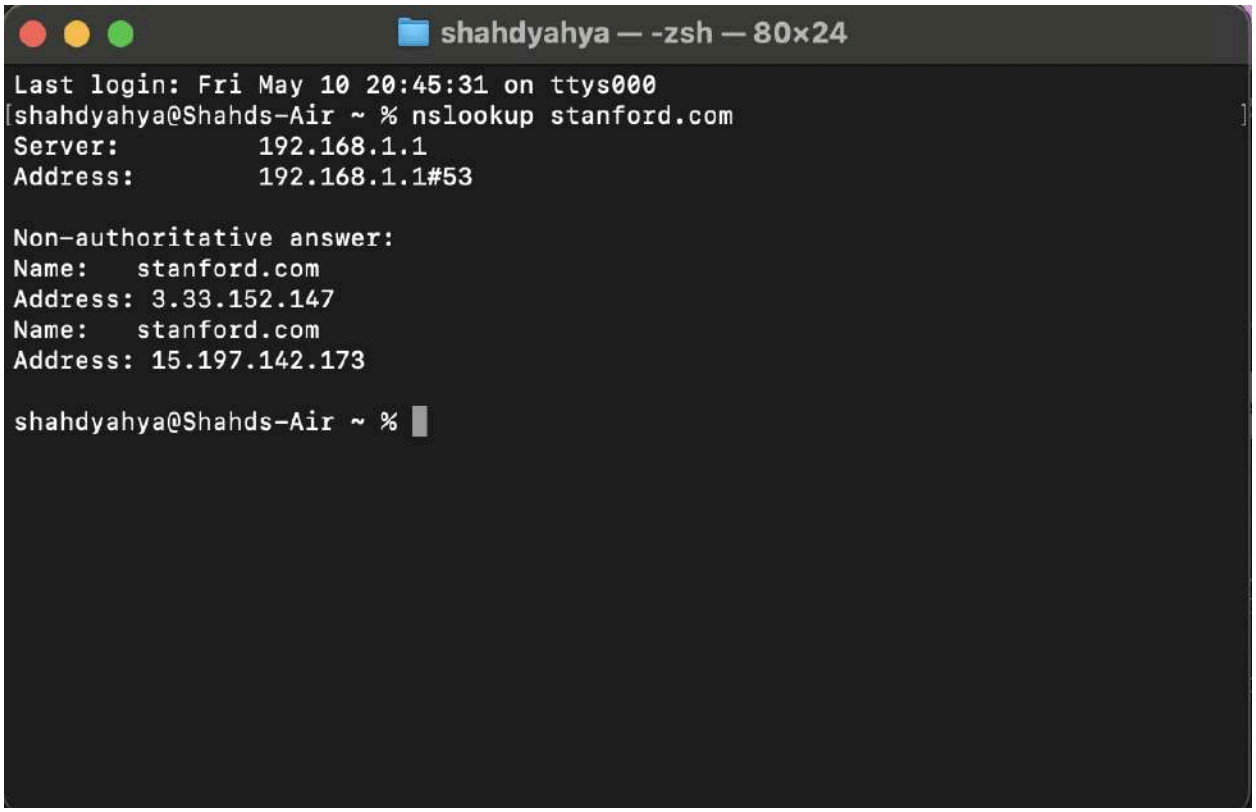
To test if the ping result is from America I put the IP address that I got(171.67.215.200) on Domain Tool on the internet and it showed that the location is America



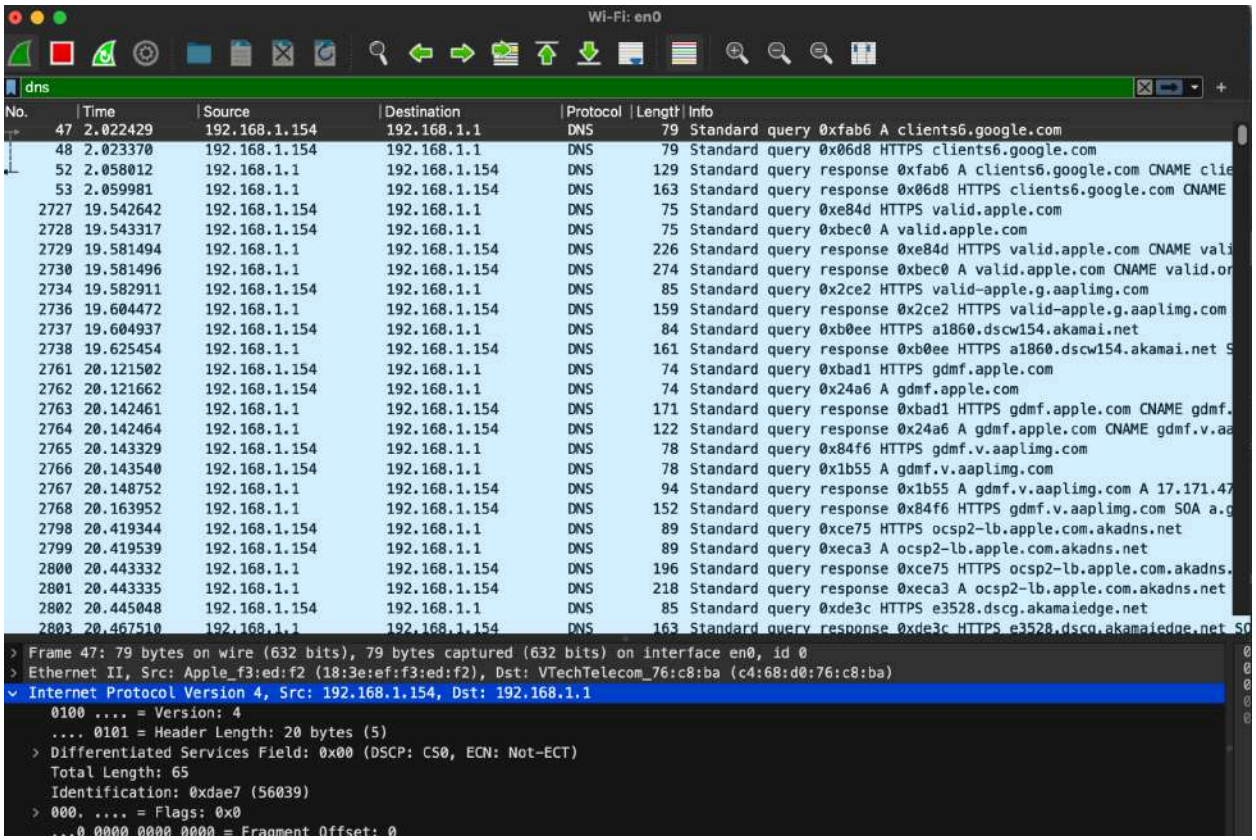
Tracert stanford.edu:



It traces the routes your data packets take to reach Stanford, listing each router (identified by IP or hostname) along the way, and it shows the time it took for a packet to reach that router and return a response, measured in milliseconds (ms).
The initial hops (1-5) are likely within your local network, showing relatively low round-trip times (around 10-70 ms), then the times increase significantly (over 200 ms), indicating the packets are traveling long distances over the internet.
The Asterisks (*) means that there was no response received from that router.
The final line (13) shows the destination (web.stanford.edu) with its IP address (171.67.215.200) and the corresponding round-trip times



I have used wireshark to capture dns results:

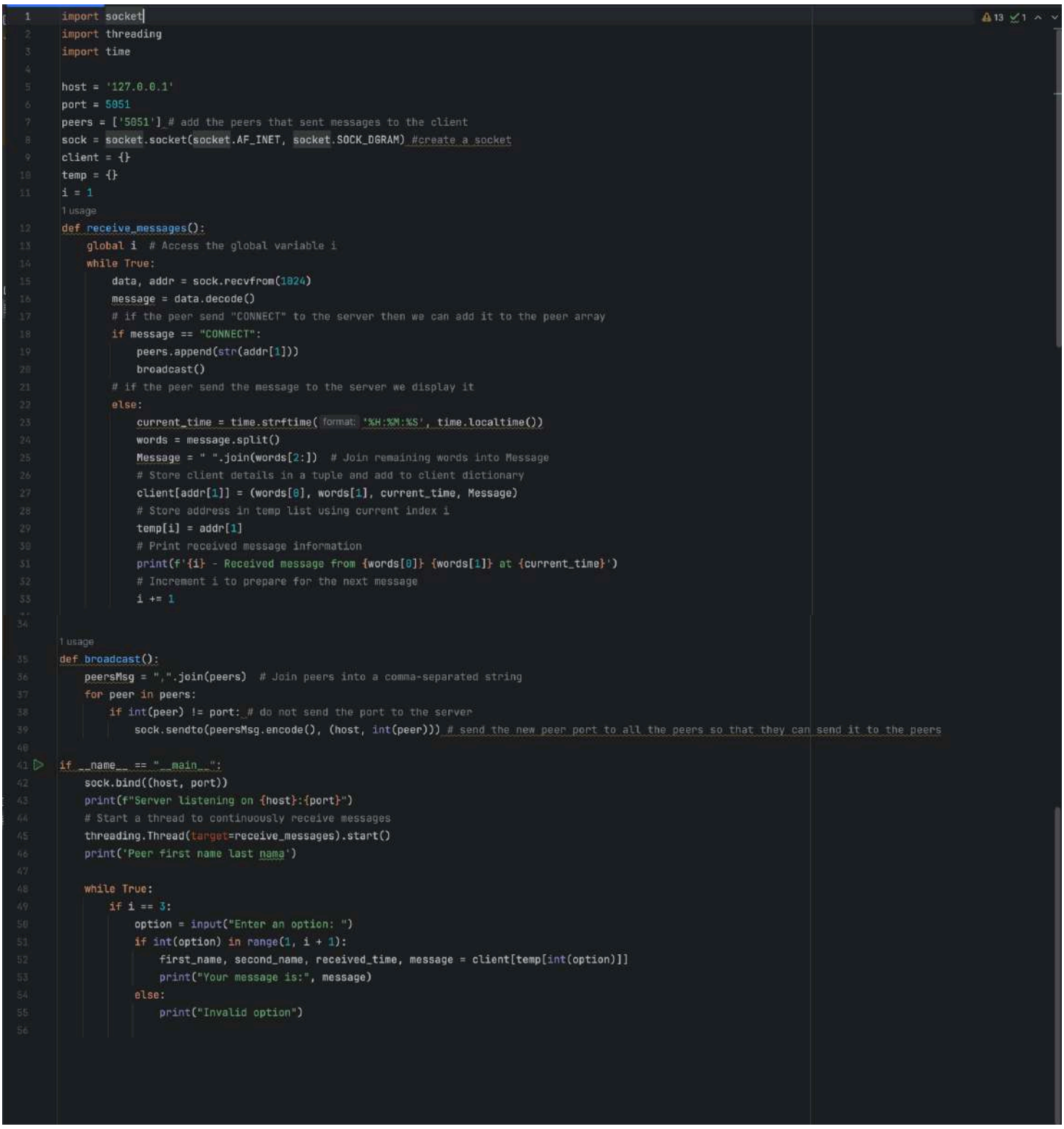


The screenshot shows a series of DNS queries made from a computer with the IP address 192.168.1.154 to a DNS server with the IP address 192.168.1.1.

Part 2:

In this part I implemented a server and a client ,the server deliver the messages from a clients and connect the peer to each other (by sending the new ports it received to the all peers so that they can send and deliver from each other)

Here is the code of the server:



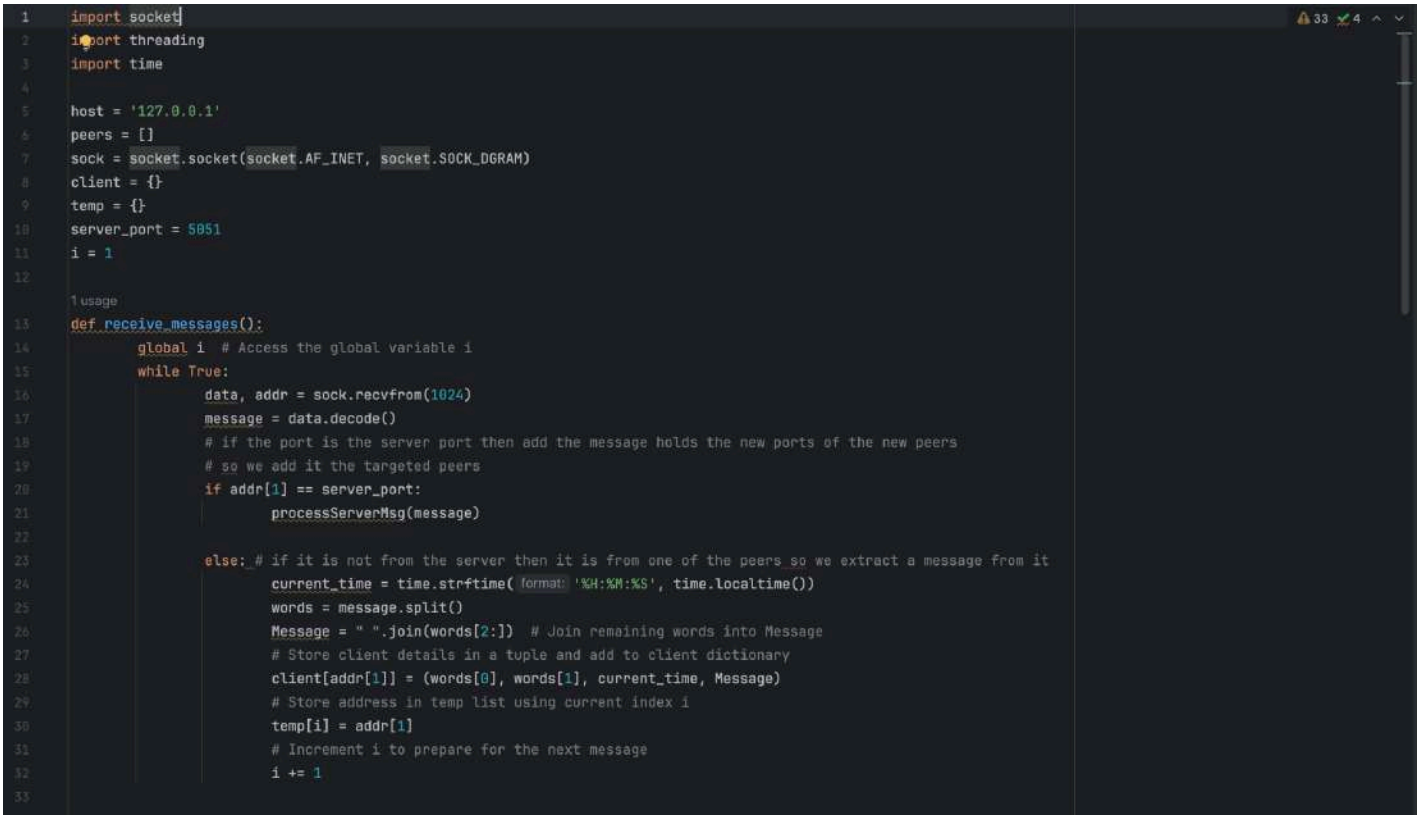
```python
import socket
import threading
import time

host = '127.0.0.1'
port = 5051
peers = ['5051'] # add the peers that sent messages to the client
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #create a socket
client = {}
temp = {}
i = 1
1 usage
def receive_messages():
    global i  # Access the global variable i
    while True:
        data, addr = sock.recvfrom(1024)
        message = data.decode()
        # if the peer send "CONNECT" to the server then we can add it to the peer array
        if message == "CONNECT":
            peers.append(str(addr[1]))
            broadcast()
        # if the peer send the message to the server we display it
        else:
            current_time = time.strftime( format: '%H:%M:%S', time.localtime())
            words = message.split()
            Message = " ".join(words[2:])  # Join remaining words into Message
            # Store client details in a tuple and add to client dictionary
            client[addr[1]] = (words[0], words[1], current_time, Message)
            # Store address in temp list using current index i
            temp[i] = addr[1]
            # Print received message information
            print(f'{i} - Received message from {words[0]} {words[1]} at {current_time}')
            # Increment i to prepare for the next message
            i += 1


1 usage
def broadcast():
    peersMsg = ",".join(peers)  # Join peers into a comma-separated string
    for peer in peers:
        if int(peer) != port: # do not send the port to the server
            sock.sendto(peersMsg.encode(), (host, int(peer))) # send the new peer port to all the peers so that they can send it to the peers

if __name__ == "__main__":
    sock.bind((host, port))
    print(f"Server listening on {host}:{port}")
    # Start a thread to continuously receive messages
    threading.Thread(target=receive_messages).start()
    print('Peer first name last nama')

    while True:
        if i == 3:
            option = input("Enter an option: ")
            if int(option) in range(1, i + 1):
                first_name, second_name, received_time, message = client[temp[int(option)]]
                print("Your message is:", message)
            else:
                print("Invalid option")
```

In this code I bind the socket to the host and port (5051). And Starts a thread to run receive_messages continuously.then it asks the user to enter first ,last name and the message.

In the threaded function it continuously listens for incoming messages using recvfrom and If the client sends "CONNECT" it adds the client's port to the peers list and broadcasts Otherwise, extracts message details, stores them in the client.

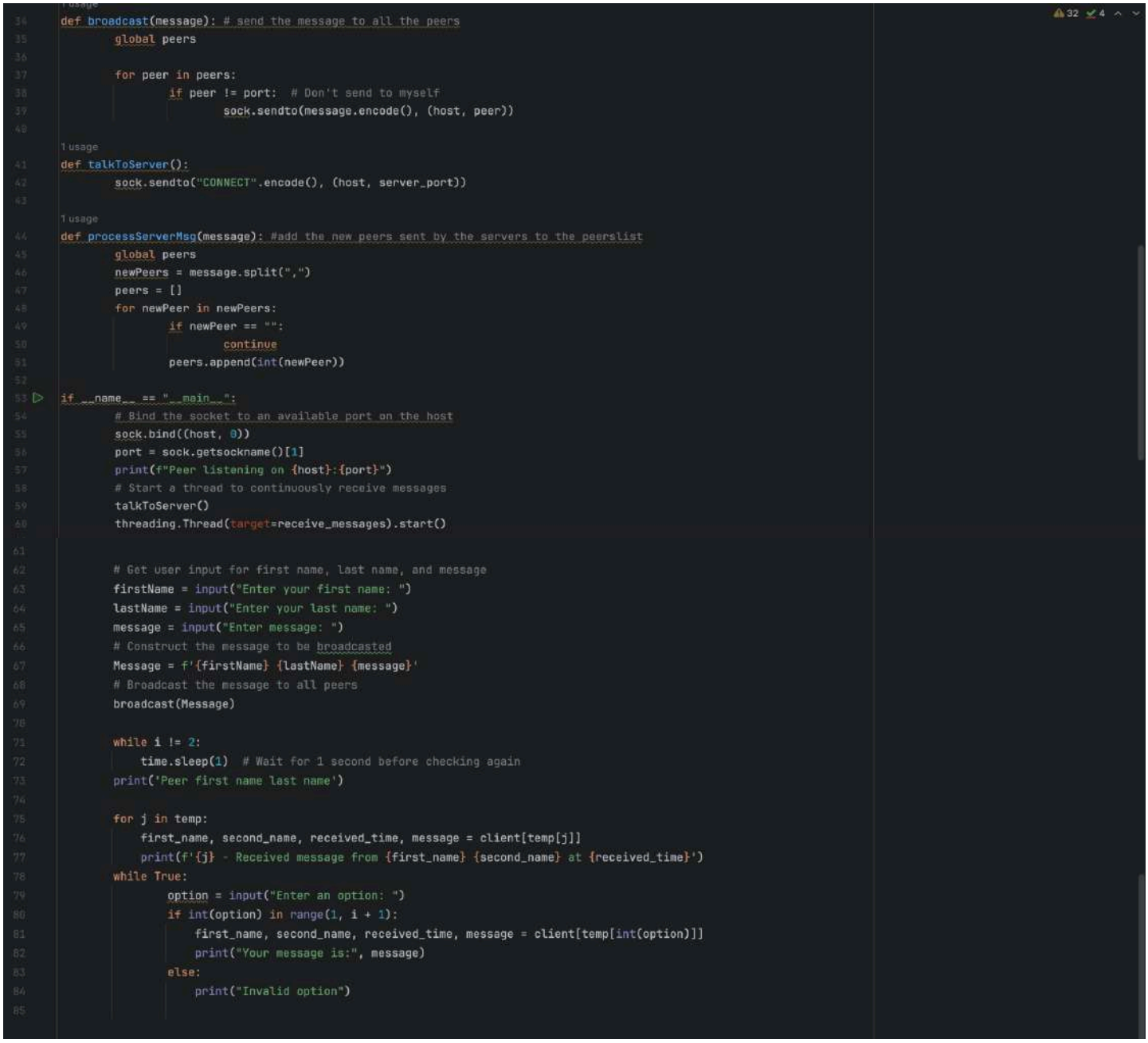
In the broadcast function it iterates through peers and sends the updated list (except the server's port) to connected clients.This allows clients to discover other connected peers.

Here is the code of the client:



```python
import socket
import threading
import time

host = '127.0.0.1'
peers = []
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client = {}
temp = {}
server_port = 5051
i = 1

1 usage
def receive_messages():
        global i  # Access the global variable i
        while True:
                data, addr = sock.recvfrom(1024)
                message = data.decode()
                # if the port is the server port then add the message holds the new ports of the new peers
                # so we add it the targeted peers
                if addr[1] == server_port:
                        processServerMsg(message)

                else: # if it is not from the server then it is from one of the peers so we extract a message from it
                        current_time = time.strftime( format: '%H:%M:%S', time.localtime())
                        words = message.split()
                        Message = " ".join(words[2:])  # Join remaining words into Message
                        # Store client details in a tuple and add to client dictionary
                        client[addr[1]] = (words[0], words[1], current_time, Message)
                        # Store address in temp list using current index i
                        temp[i] = addr[1]
                        # Increment i to prepare for the next message
                        i += 1
```

The rest of the code :



```
def broadcast(message): # send the message to all the peers
        global peers

        for peer in peers:
                if peer != port:  # Don't send to myself
                        sock.sendto(message.encode(), (host, peer))

1 usage
def talkToServer():
        sock.sendto("CONNECT".encode(), (host, server_port))

1 usage
def processServerMsg(message): #add the new peers sent by the servers to the peerslist
        global peers
        newPeers = message.split(",")
        peers = []
        for newPeer in newPeers:
                if newPeer == "":
                        continue
                peers.append(int(newPeer))

if __name__ == "__main__":
        # Bind the socket to an available port on the host
        sock.bind((host, 0))
        port = sock.getsockname()[1]
        print(f"Peer listening on {host}:{port}")
        # Start a thread to continuously receive messages
        talkToServer()
        threading.Thread(target=receive_messages).start()


        # Get user input for first name, last name, and message
        firstName = input("Enter your first name: ")
        lastName = input("Enter your last name: ")
        message = input("Enter message: ")
        # Construct the message to be broadcasted
        Message = f'{firstName} {lastName} {message}'
        # Broadcast the message to all peers
        broadcast(Message)

        while i != 2:
            time.sleep(1)  # Wait for 1 second before checking again
        print('Peer first name last name')

        for j in temp:
            first_name, second_name, received_time, message = client[temp[j]]
            print(f'{j} - Received message from {first_name} {second_name} at {received_time}')
        while True:
                option = input("Enter an option: ")
                if int(option) in range(1, i + 1):
                    first_name, second_name, received_time, message = client[temp[int(option)]]
                    print("Your message is:", message)
                else:
                    print("Invalid option")
```

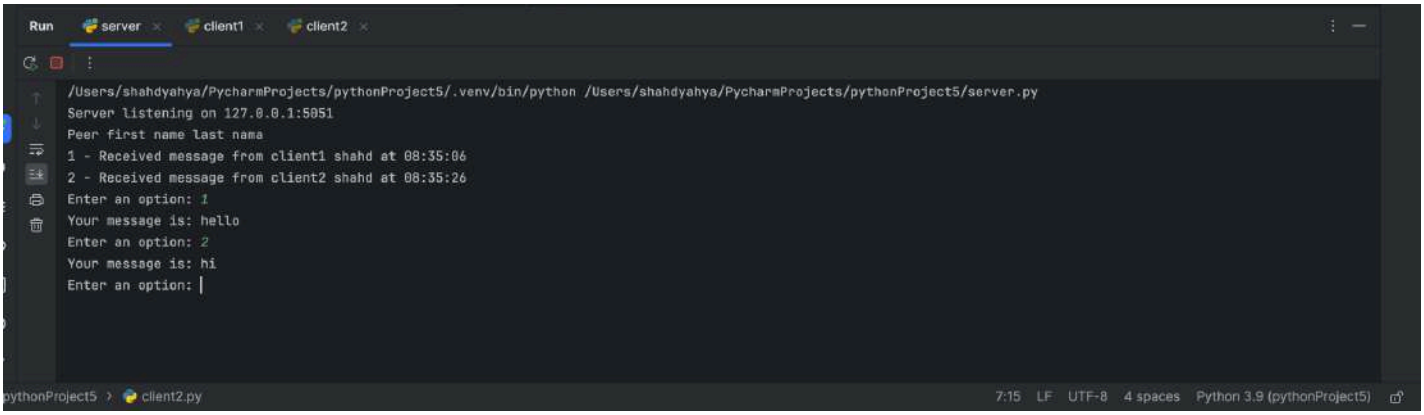
In this code I bind the socket to an available port on the local machine.and then starts a thread for receive_messages to listen for messages continuously.then Prompts the user to enter their name and a message.and Broadcasts the message to all connected peers using the broadcast function.

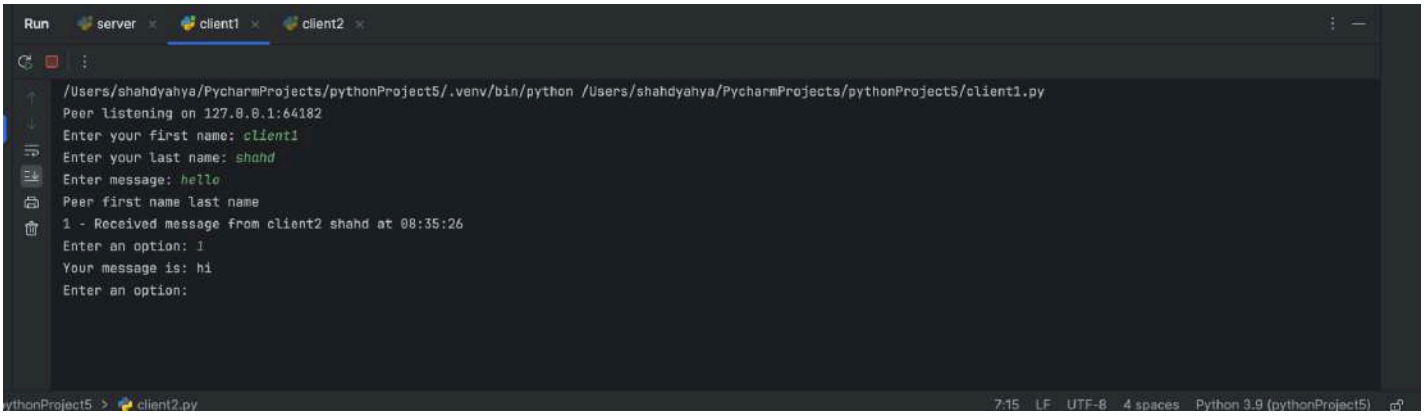
In the threaded function receive_messages it listens for incoming messages.If the message comes from the server port (5051), it means that the server sent a new peer port and calls processServerMsg to update the list of connected peers .Otherwise, it extracts message details.

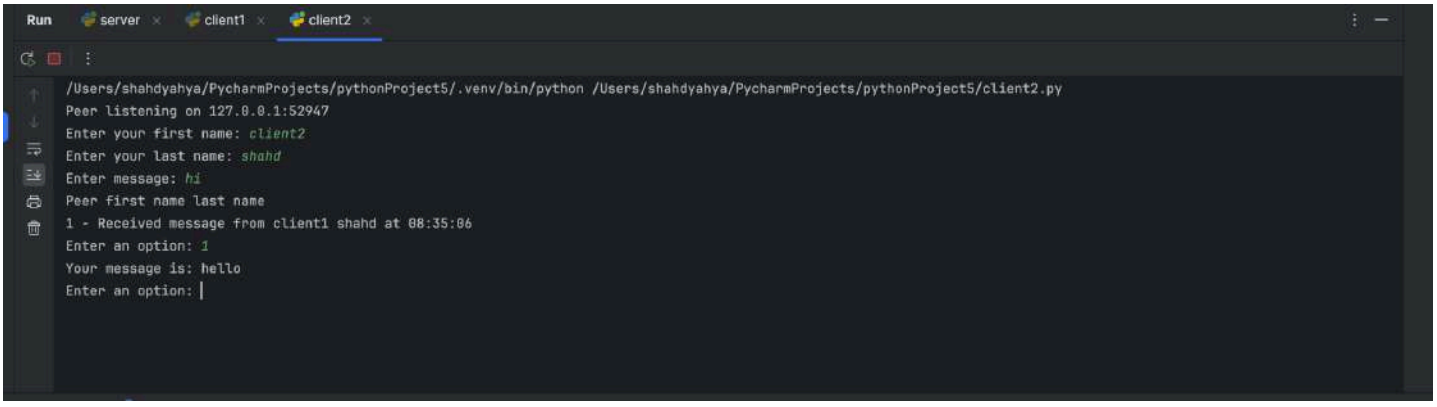
In the broadcast function it iterates through the peers list and sends the message to each connected client.

In the talkToServer function it sends a "CONNECT" message to the server to announce the client's presence.

Here is the output of a server and 2 clients:







Part 3:

Entity tag cache validators : are unique identifiers that act as a more dependable way to validate cached content. This improves efficiency by letting browsers reuse cached content when it truly hasn't changed, reducing traffic and server load.

Here is the code of my program:

```python
from socket import *

serverPort = 6060
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while True:
    connectionSocket, addr = serverSocket.accept()
    ip = addr[0]
    port = addr[1]
    sentence = connectionSocket.recv(1024).decode()
    request = sentence.split()[1]
    print("the request is : " + request)
    if (request == '/' or request == '/index.html' or request == '/main_en.html' or request == '/en'): # The if statement checks whether the requested object is
        connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
        connectionSocket.send("Content-Type: text/html \r\n".encode())
        connectionSocket.send("\r\n".encode())
        fileminen = open("main_en.html", "rb")
        connectionSocket.send(fileminen.read())  # read the file that was open  when it called

    elif (request == '/ar'): # The if statement checks whether the requested object is one of several specific values
        connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
        connectionSocket.send("Content-Type: text/html \r\n".encode())
        connectionSocket.send("\r\n".encode())
        fileminen = open("main_ar.html", "rb")
        connectionSocket.send(fileminen.read())  # read the file that was open  when it called

    elif (request.endswith('.html')):
        connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
        connectionSocket.send("Content-Type: text/html \r\n".encode())
        connectionSocket.send("\r\n".encode())
        filelink = open("myform.html", "rb")
        connectionSocket.send(filelink.read())

    elif (request.endswith('.css')):
        connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
        connectionSocket.send("Content-Type: text/css \r\n".encode())
        connectionSocket.send("\r\n".encode())
        filecss = open("style.css", "rb")
        connectionSocket.send(filecss.read())

    elif (request.endswith('.png')):  # files with the extensions '.png' and '.jpg'.
        connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
        connectionSocket.send("Content-Type: image/png \r\n".encode())
        connectionSocket.send("\r\n".encode())
        filepngimg = open("image3.png", "rb")
        connectionSocket.send(filepngimg.read())

    elif (request.endswith('.jpg')):  # The same process occurs for '.jpg' files,
        connectionSocket.send("HTTP/1.1 200 OK \r\n".encode())
        connectionSocket.send("Content-Type: image/jpg \r\n".encode())
        connectionSocket.send("\r\n".encode())
        filejpgimg = open("image4.jpg", "rb")  # open the image with jpg extension.
        connectionSocket.send(filejpgimg.read())

    elif (request == '/so'):

        connectionSocket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
        connectionSocket.send("Content-Type: text/html \r\n".encode())
        connectionSocket.send("Location: https://stackoverflow.com \r\n".encode())
        connectionSocket.send("\r\n".encode())
```

The rest of the code:

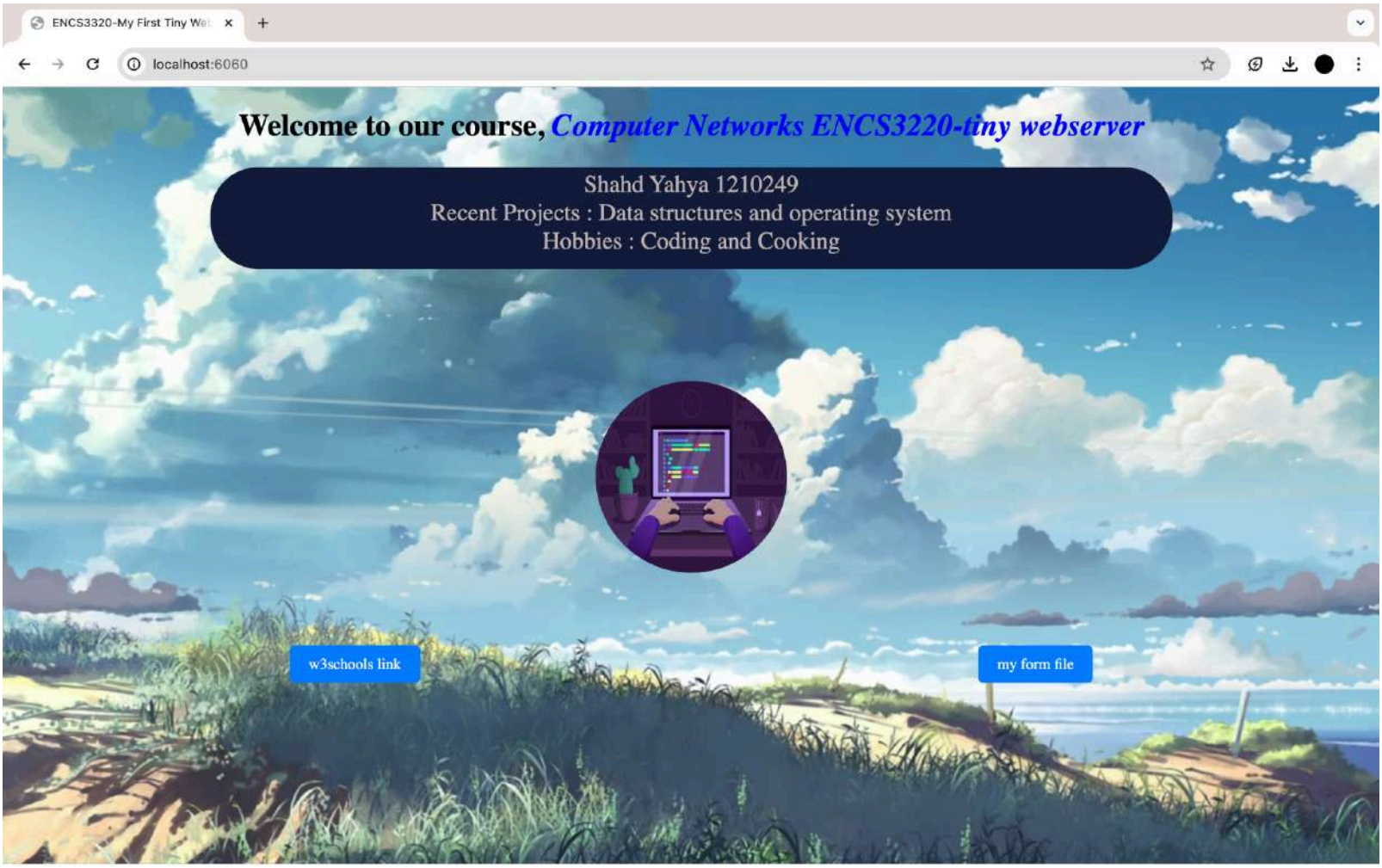
```
elif (request == '/itc'):

    connectionSocket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
    connectionSocket.send("Content-Type: text/html \r\n".encode())
    connectionSocket.send("Location: https://itc.birzeit.edu \r\n".encode())
    connectionSocket.send("\r\n".encode())

else:  # scenario where a requested resource is not found by a client.
    connectionSocket.send("HTTP/1.1 404 Not Found \r\n".encode())
    connectionSocket.send("Content-Type: text/html \r\n".encode())
    connectionSocket.send("\r\n".encode())
    notFoundHtmlString = "<html>" \
                         "<head>" \
                         "<title>ERROR 404 </title>" \
                         "</head>" \
                         "<body>" \
                         "<pre>" \
                         "<p style ='font-size: 45px; text-align:center; color:Red'>" \
                         "The file is not found </p>" \
                         "<p style ='font-size: 25px; text-align:center; color:Black'>" \
                         "<b> Shahd Yahya 1210249 <b/></p>" \
                         "</p>" \
                         "<pre style='font-size: 25px; text-align:center;'>" \
                         f"The IP is: {ip}     " \
                         f"The port is: {port}" \
                         "</p>" \
                         "</body>" \
                         "</html>"
    notFoundHtmlBytes = bytes(notFoundHtmlString, "UTF-8")
    connectionSocket.send(notFoundHtmlBytes)
```

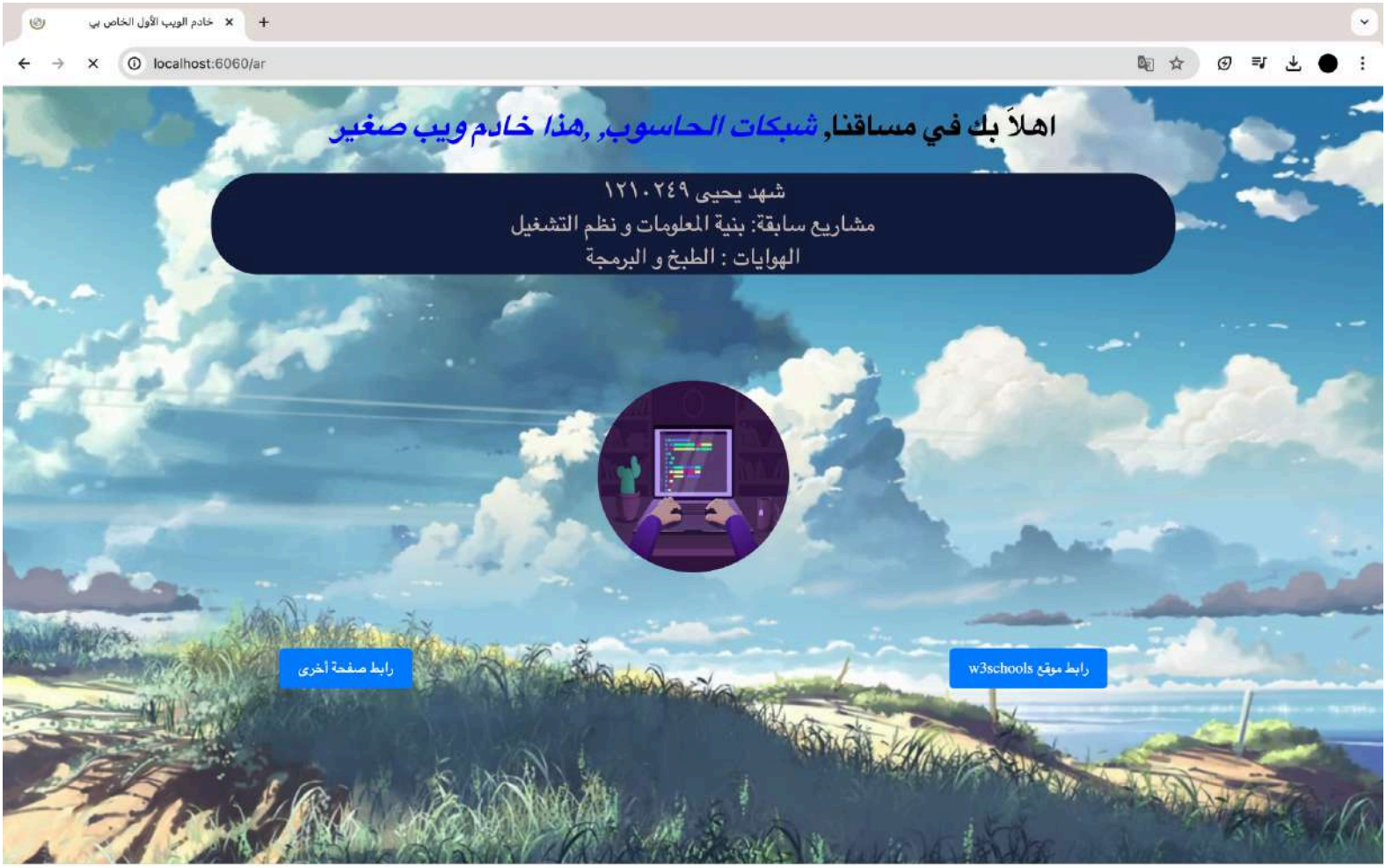
First I defined the server port (6060).then created a TCP socket (serverSocket) for listening on this port.then binds the socket to the port and starts listening for incoming connections.
When a client connects, serverSocket.accept() accepts the connection and returns a new socket and it receives the request message sent by the client and decodes it.

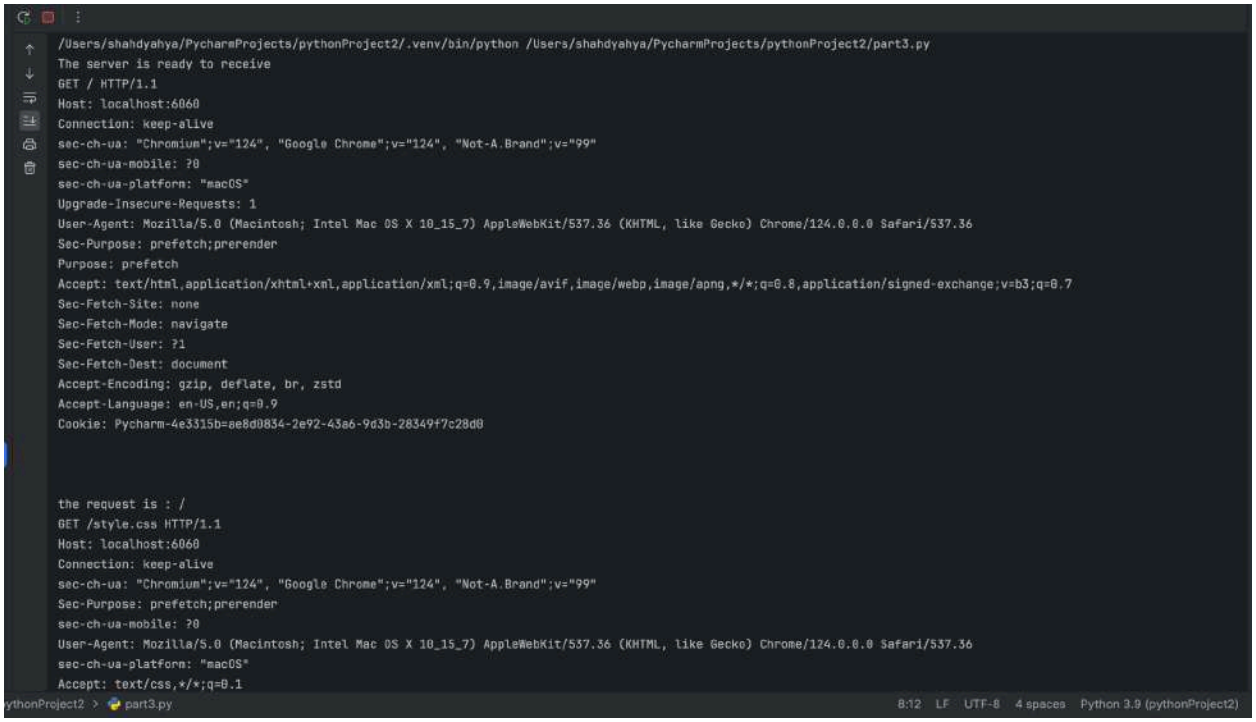
- If the requested object is main_en.html, index.html, or similar, it sends a 200 OK response with the content of "main_en.html".
- If the requested object ends with ".html" (but isn't a specific file mentioned before), it sends a 200 OK response with the content of "myform.html".
- If it ends with ".css", it sends a 200 OK response with the content of "style.css" (assuming a CSS file).
- If it ends with ".png", it sends a 200 OK response with the content of "image3.png" (assuming a PNG image).
- If it ends with ".jpg", it sends a 200 OK response with the content of "image4.jpg" (assuming a JPG image).
- If the requested object is "/so", it sends a 307 Temporary Redirect response to redirect the client to Stack Overflow.

- If the requested object is "/itc", it sends a 307 Temporary Redirect response to redirect the client to Birzeit University.
- If none of the above match (file not found), it sends a 404 Not Found response with a custom error message that includes the client's IP and port.

The web server :




خادم الويب الأول الخاص بي
localhost:6060/ar
اهلاً بك في مساقنا, شبكات الحاسوب, ,هذا خادم ويب صغير
شهد يحيى ١٢١٠٢٤٩
مشاريع سابقة: بنية المعلومات و نظم التشغيل
الهوايات : الطبخ و البرمجة
رابط صفحة أخرى
رابط موقع w3schools


/Users/shahdyahya/PycharmProjects/pythonProject2/.venv/bin/python /Users/shahdyahya/PycharmProjects/pythonProject2/part3.py
The server is ready to receive
GET / HTTP/1.1
Host: localhost:6060
Connection: keep-alive
sec-ch-ua: "Chromium";v="124", "Google Chrome";v="124", "Not-A.Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36
Sec-Purpose: prefetch;prerender
Purpose: prefetch
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9
Cookie: Pycharm-4e3315b=ae8d0834-2e92-43a6-9d3b-28349f7c28d0
the request is : /
GET /style.css HTTP/1.1
Host: localhost:6060
Connection: keep-alive
sec-ch-ua: "Chromium";v="124", "Google Chrome";v="124", "Not-A.Brand";v="99"
Sec-Purpose: prefetch;prerender
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36
sec-ch-ua-platform: "macOS"
Accept: text/css,*/*;q=0.1
ythonProject2 > part3.py
8:12 LF UTF-8 4 spaces Python 3.9 (pythonProject2)


ERROR 404
localhost:6060/grtgrt
The file is not found
Shahd Yahya 1210249
The IP is: 127.0.0.1 The port is: 54670


ENCS3320-My Tiny Webserv
localhost:6060/myform.html
Image name image3.png
Open Image

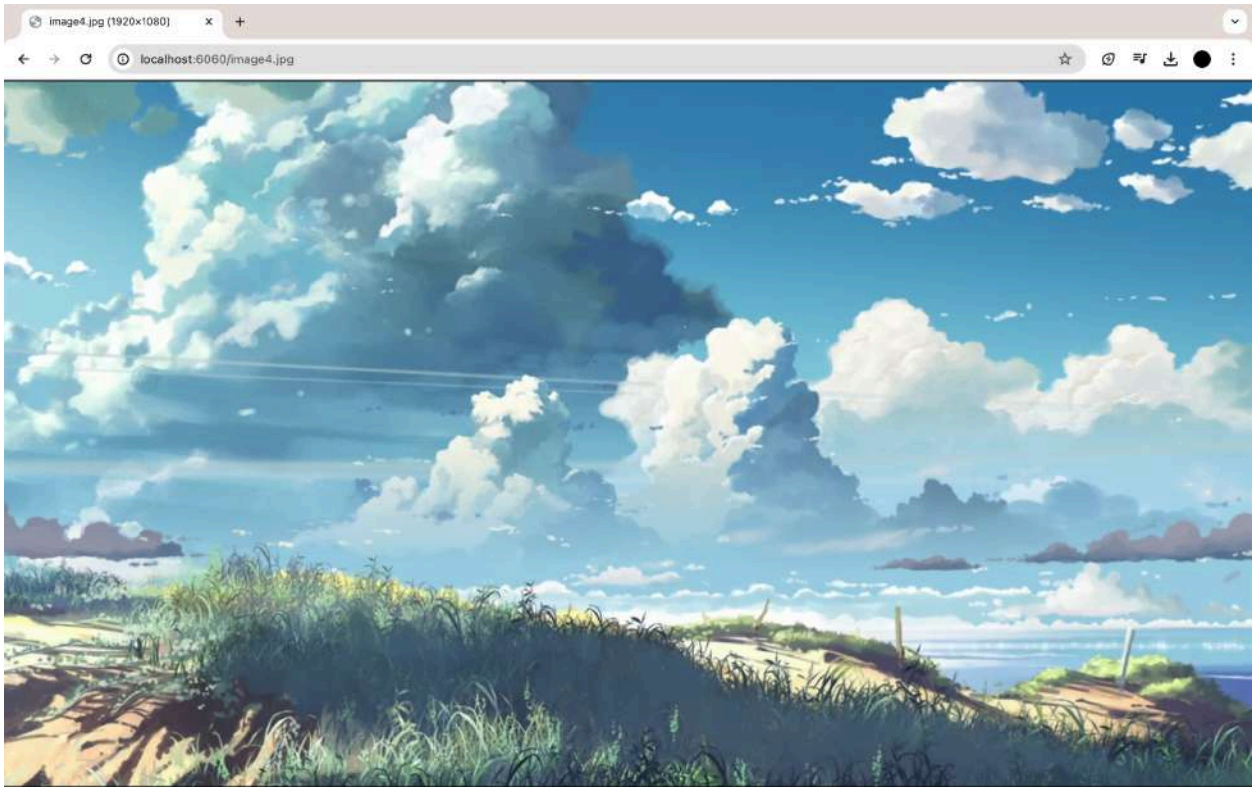


image4.jpg (1920×1080)
localhost:6060/image4.jpg

image3.png (421×296)

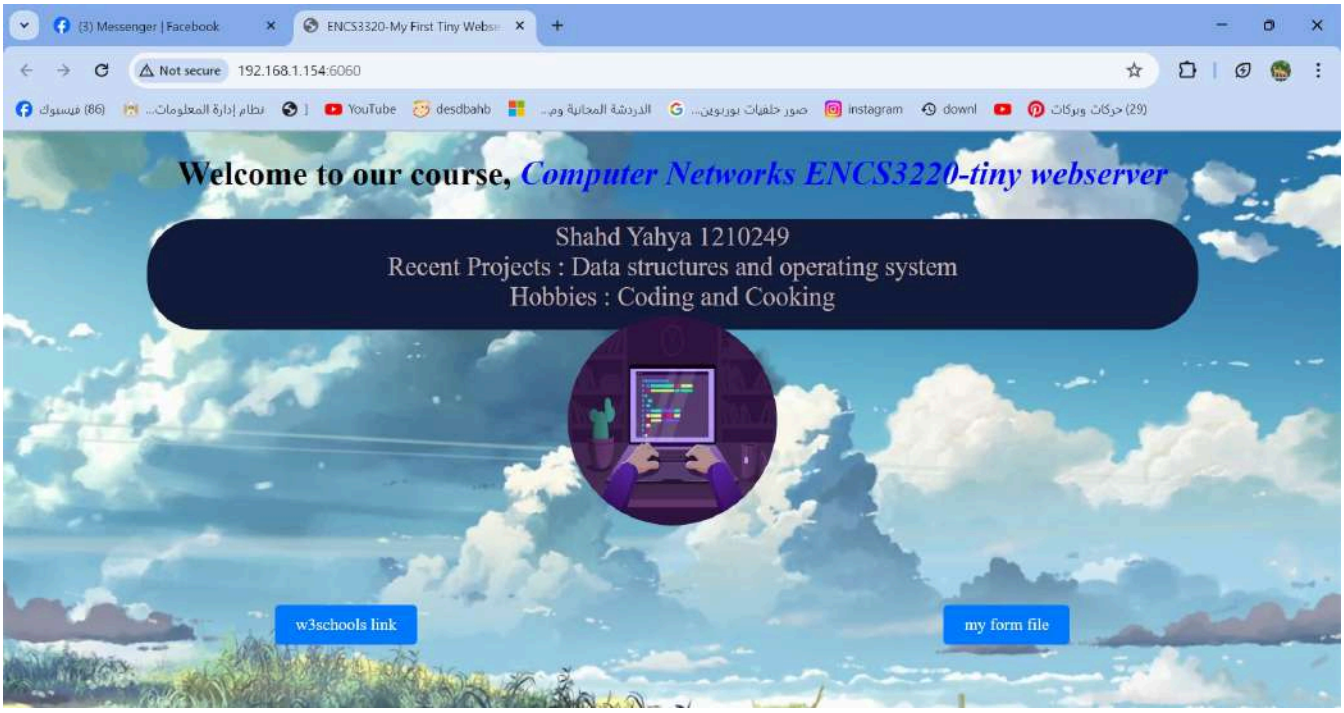
localhost:6060/image3.png

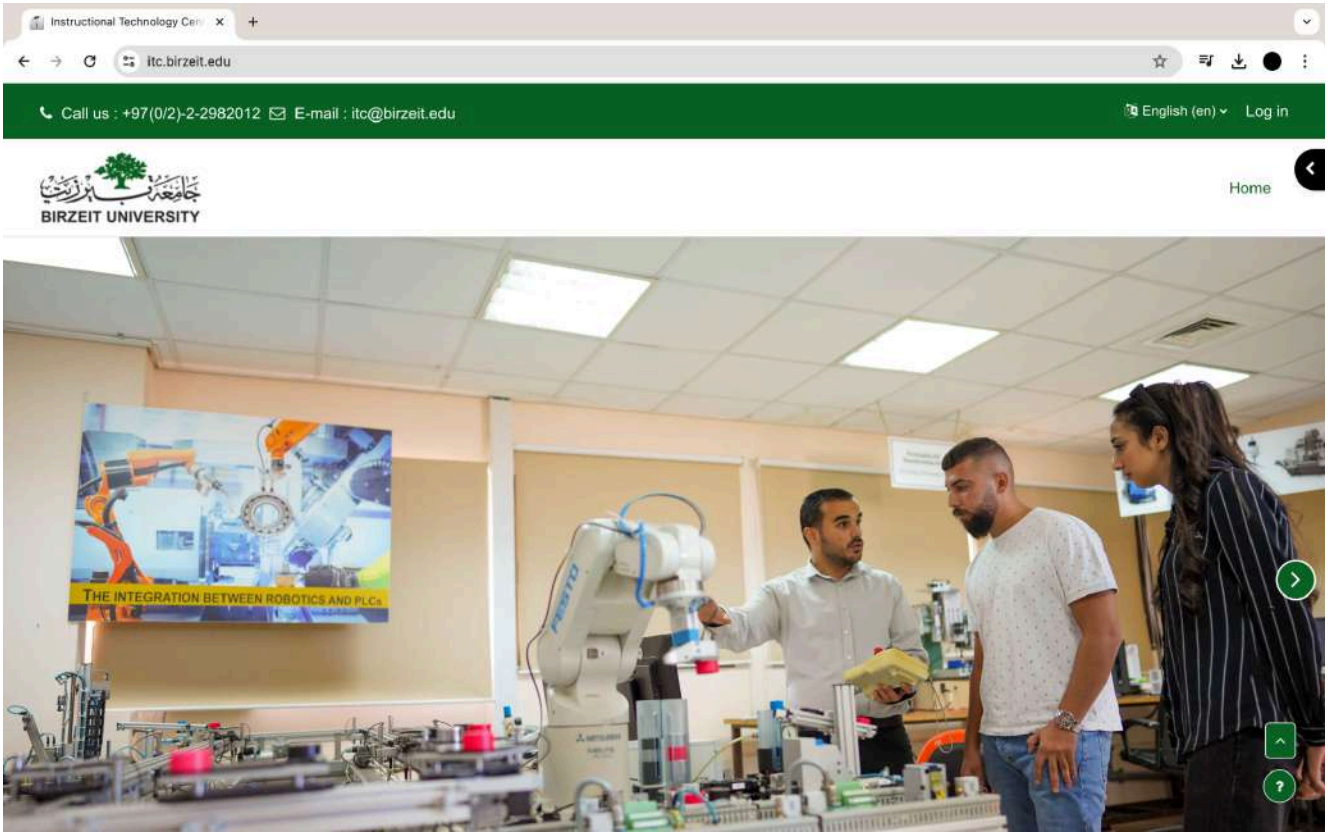
localhost:6060/main.css

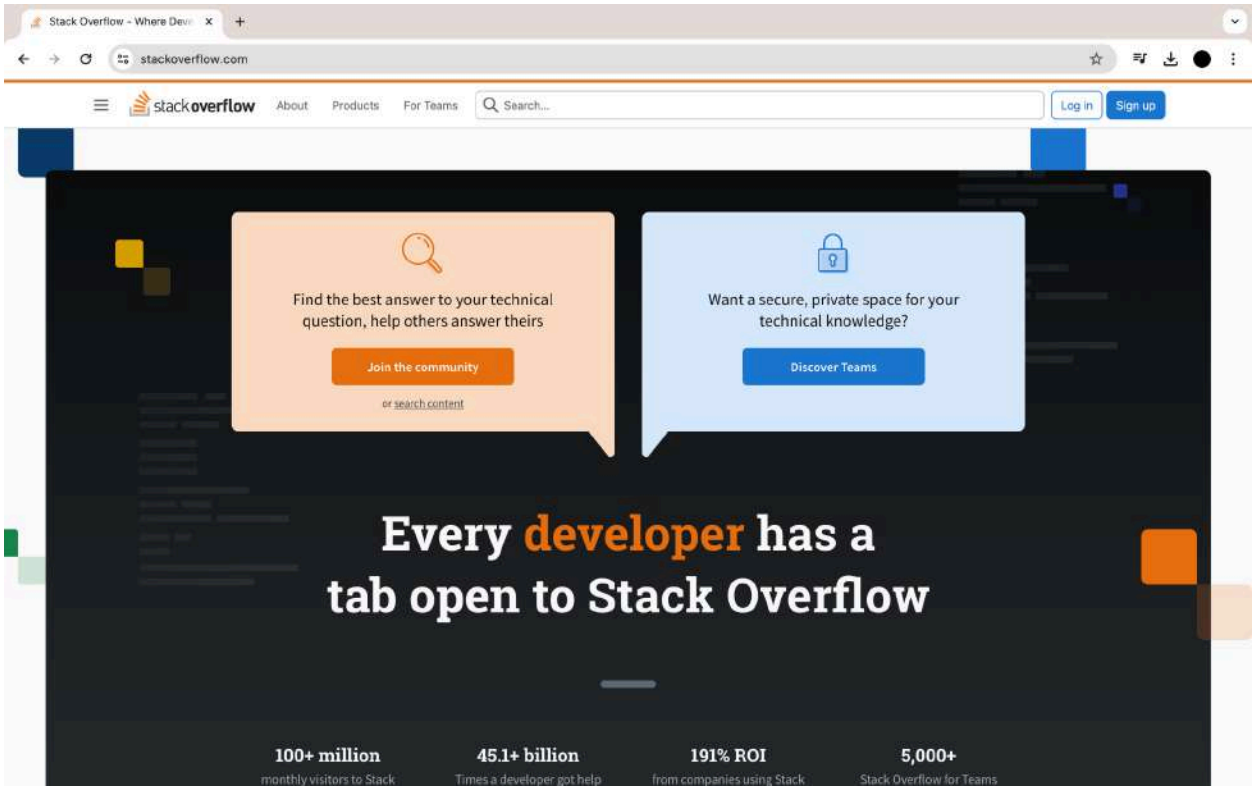
localhost:6060/main.css

```
  body {/* Background pic */
    background-image: url("image4.jpg");
    background-position: center top;
    background-repeat: no-repeat;
    background-size: cover;
}
  h1 {
      text-align: center;
        }
  em {
      color : blue;
        }

  .container1 {
      background-color: #131d3b;/* boxes of names and hobbies*/
      border-radius:50px;
      color: #c2b3b3;
      font-family:serif;
      font-size:130%;
      padding:3px;
      margin:25px  auto;
      width: 1000px;
      height: 100px;
      text-align: center;


        }
  p{
      font-size: 25px;
      margin: 0 auto;

  }

  img{
      width: 200px;
      height: 200px;
      border-radius: 100px;
      position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%); /* Center the image using transform */


        }
 .container2 {
    display: flex;
    justify-content: center;
    align-items: flex-end;
    min-height: 50vh;
    padding-bottom: 20px;
}

a.a, a.b {
```

From other device:






Stack Overflow - Where Deve
stackoverflow.com
stackoverflow
About
Products
For Teams
Search...
Log in
Sign up
Find the best answer to your technical question, help others answer theirs
Join the community
or search content
Want a secure, private space for your technical knowledge?
Discover Teams
Every developer has a tab open to Stack Overflow
100+ million
monthly visitors to Stack
45.1+ billion
Times a developer got help
191% ROI
from companies using Stack
5,000+
Stack Overflow for Teams